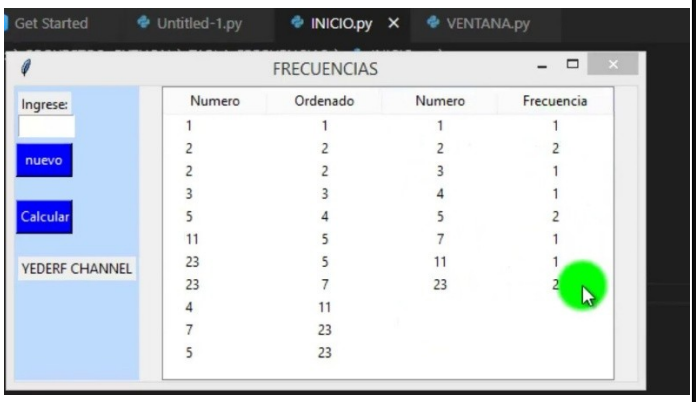
1. Estadística Descriptiva con Python

1.1 Estadística Descriptiva

La estadística descriptiva resume conjuntos de datos mediante valores como media, mediana, moda, desvia

1.2 Código con explicació




```python
import numpy as np

import matplotlib.pyplot as plt

from scipy import stats


# Generar datos aleatorios

datos = np.random.normal(loc=50, scale=10, size=1000)


# Calcular estadísticas

media = np.mean(datos)

mediana = np.median(datos)

moda = stats.mode(datos, keepdims=True)[0][0]

desviacion = np.std(datos)

varianza = np.var(datos)


# Mostrar resultados

Print(f"Media: {media:.2f}")

Print(f"Mediana: {mediana:.2f}")

print(f"Moda: {moda:.2f}")

print(f"Desviación estándar: {desviacion:.2f}")

print(f"Varianza: {varianza:.2f}")


# Histograma

plt.hist(datos, bins=30, color='skyblue', edgecolor='black')

plt.title("Histograma de datos simulados")

plt.xlabel("Valor")

plt.ylabel("Frecuencia")

plt.grid(True)
```

1.3 Aplicación cotidiana:

Evaluar el rendimiento de estudiante

analizar precios de productos,

interpretar distribuciones de

mediciones físicas.

2. Cálculo de Límites con SymPy

2.1 Cálculo de Límites

Un límite describe el comportamiento de una función cuando se acerca a un valor específico.

2.2 Código:

```
import sympy as sp


x = sp.symbols('x')

func = (sp.sin(x) / x)


# Calcular el límite cuando x → 0

limite = sp.limit(func, x, 0)


print("Límite de sin(x)/x cuando x → 0:",
limite)
```

2.3 Resultado esperado:

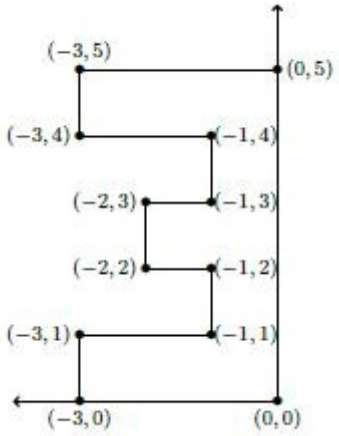
```
Límite de sin(x)/x cuando x → 0:
1
```

2.4 Aplicación:

Se usa en física (velocidades instantáneas), en ingeniería (derivadas), y economía (tasas de cambio).

3. Álgebra Lineal con NumPy

Inversas, determinantes y sistemas de ecuaciones




```python
import numpy as np

# Matriz cuadrada
A = np.array([[3, 2], [1, 4]])

# Inversa
A_inv = np.linalg.inv(A)
print("Inversa de A:\n", A_inv)

# Determinante
det = np.linalg.det(A)
print("Determinante:", det)

# Resolver Ax = b
b = np.array([5, 6])
x = np.linalg.solve(A, b)
print("Solución del sistema Ax=b:", x)
```

Aplicaciones:

Modelado de circuitos eléctricos

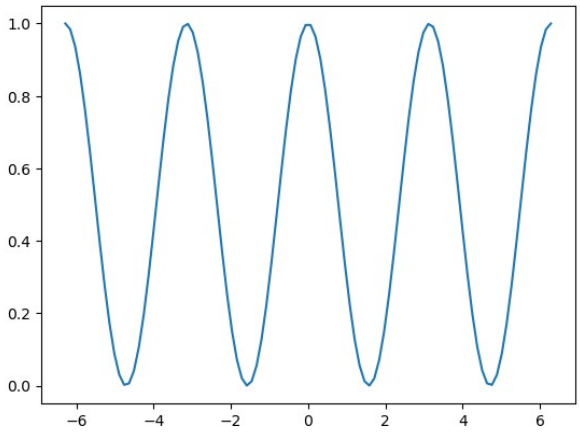
Resolución de fuerzas físicas

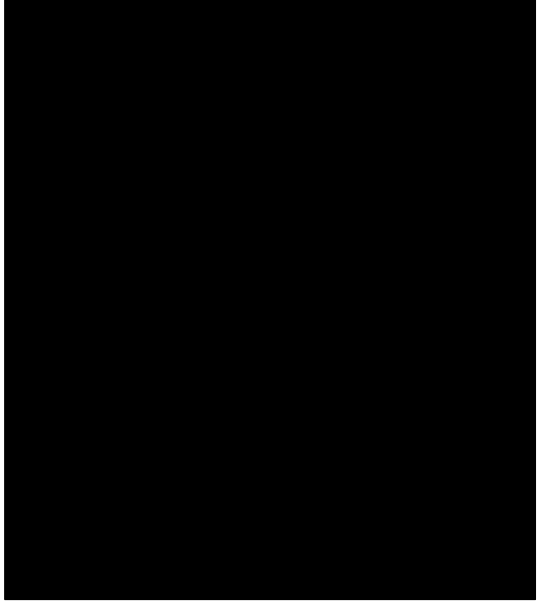
Economía y análisis de portafolios

4. Gráficas de funciones 2D y 3D

4.1 Gráficas 2D

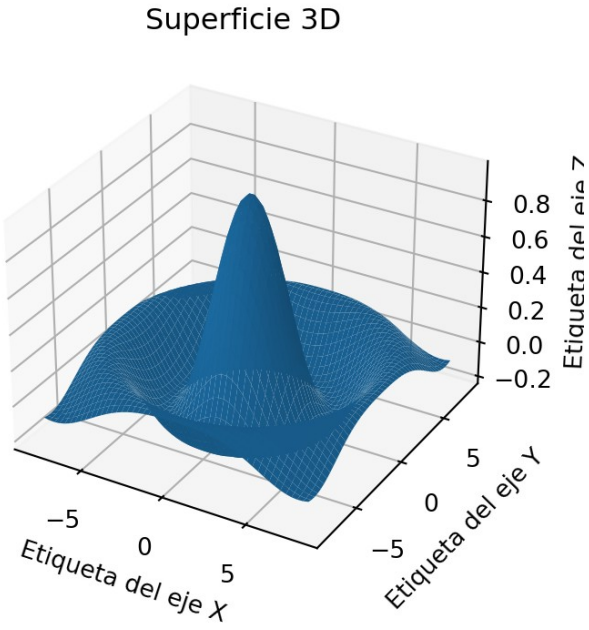
```python
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-10, 10, 400)
y = np.sin(x)
```





4.2 Gráficos 3D

```
from mpl_toolkits.mplot3d
import Axes3D


x = np.linspace(- 5, 5, 50)

y = np.linspace(- 5, 5, 50)

X, Y = np.meshgrid(x, y)

Z = np.sin(np.sqrt(X**2 +
Y**2))


fig = plt.figure()

ax = fig.add_subplot(111,
projection='3d')

ax.plot_surface(X, Y, Z,
cmap='viridis')

ax.set_title("Gráfico 3D: z =
sin(√(x² + y²))")

plt.show()
```



ráficas, ondas, funciones de muchas

5. Optimización: Mínimos y Máximos

Ejemplo: Función con mínimo

```python
import sympy as sp

x = sp.symbols('x')

f = x**4 - 4*x**2 + 3

# Derivar la función

f_prime = sp.diff(f, x)

# Encontrar los puntos críticos

criticos = sp.solve(f_prime, x)

# Verificar máximos o mínimos

for c in criticos:

    segunda_derivada = sp.diff(f_prime, x).subs(x, c)

    tipo = "mínimo" if segunda_derivada > 0 else "máximo"

    print(f"x = {c} es un {tipo}")
```

Aplicaciones:

Optimizar beneficios (economía)

Minimizar tiempo o materiales (ingeniería)

Encontrar puntos de equilibrio (física)

NOTA : Si no funcionan los códigos, puede que no tengas las librerías necesarias para abrir el programa te doy el siguiente link con videos relacionados con librerías :

https://youtu.be/SWIIQboGGCQ